

Combinatorial ranking algorithms for the general linear group

Gustavo Zambonin Tiago Siqueira Larissa Gremelmaier
Rosa Rodrigo Stefanos Ricardo Custódio Daniel Panario

Universidade Federal de Santa Catarina & Carleton University

April 16, 2026

LATIN 2026

The (combinatorial) object in question

Definition

The $n \times n$ invertible matrices over $\mathbb{F}_q$ equipped with the usual multiplication form the *general linear group* $\text{GL}_n(q)$.

Theorem

$$|\text{GL}_n(q)| = \prod_{i=0}^{n-1} (q^n - q^i).$$

In quantum-safe cryptography, $\text{GL}_n(q)$ appears in protocols based on the *matrix code equivalence* problem (e.g., MEDS).

Uniform sampling from the group is quite common (also in linear algebra and computational group theory, as $\text{Aut}(\mathbb{F}_q^n) \cong \text{GL}_n(q)$).

Heuristic methods in the cryptography literature (additive repair, *PLU* construction) do *not* sample uniformly from $\text{GL}_n(q)$.

Algorithms to sample from $GL_n(q)$

Rejection sampling is often used: draw a matrix M from $M_n(\mathbb{F}_q)$, reject if $\det(M) = 0$, and try again until the condition is met.

The number of trials is $1/\prod_{i=1}^n (1 - q^{-i})$ on average, which approaches 1 as $q \rightarrow \infty$. (But q is usually smaller.)

Some alternatives: *subgroup* algorithm

[Diaconis and Shahshahani, 1987], *NonSing* [Randall, 1993], using *Bruhat* decomposition or counting *Schubert* cells [Pak, 1997].

We focus on the *random bit complexity* of these methods, which is either not optimal, or not trivial to implement (Schubert).

Algorithms to sample from $\text{GL}_n(q)$

Combinatorial (un)ranking as an alternative

Definition

Let S be a finite set and $N = |S|$. A *ranking function* is a bijection $\mathbf{R} : S \rightarrow [N]$ that defines a total order $\prec$ on the elements of S .

The *unranking function* is the inverse bijection $\mathbf{U} = \mathbf{R}^{-1}$, i.e., $\mathbf{U} : [N] \rightarrow S$ such that $\mathbf{R}(s) = i$ iff $\mathbf{U}(i) = s$ for all $i \in [N]$.

To draw a uniform random element x of S , sample exactly $\lceil \lg |\text{GL}_n(q)| \rceil$ random bits, and evaluate $s \leftarrow \mathbf{U}(x)$.

Cryptographic protocols may also benefit from being able to compress s losslessly via $\mathbf{R}(s)$, reducing transmission overhead.

We could not find combinatorial ranking algorithms for $\text{GL}_n(q)$, so we made our own.

The “combinatorial tower” intuition

Ranking field elements and vectors

The simplest components of $\text{GL}_n(q)$ are its entries and columns. We can (un)rank these by attaching a (reverse) lexicographic order.

For $q = p^d$, represent $f \in \mathbb{F}_q$ as a polynomial $\sum_{i=0}^{d-1} a_i x^i$. Ranking is the standard base- p evaluation $\mathbf{R}_P(f) = \sum_{i=0}^{d-1} a_i p^i$.

For $v = \sum_{i=0}^{n-1} v_i b_i \in \mathbb{F}_q^n$, map each coordinate to its scalar rank $k_i = \mathbf{R}_P(v_i)$, relative to an ordered basis $B = \{b_0, \dots, b_{n-1}\}$. Ranking is the base- q evaluation $\mathbf{R}_V(B, v) = \sum_{i=0}^{n-1} k_i q^i$.

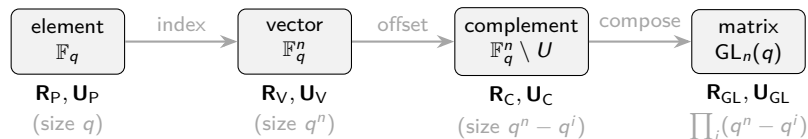
Unranking is the reverse operation: *finding* the specific sequence of coefficients that reconstructs the target rank.

The “combinatorial tower” intuition

Ranking invertible matrices

The $|\mathrm{GL}_n(q)|$ formula suggests that we should choose among the first $q^n - q^0$ vectors, then the $q^n - q^1$ remaining etc.

In other words, this is a mixed-radix system with radices $\mathbf{q} = (q^n - q^0, \dots, q^n - q^{n-1})$, generalizing the “single” radix p or q .



To compute $\mathbf{R}_{\mathrm{GL}}$, we must evaluate $\mathbf{R}_{\mathrm{C}}$ given previous columns.

In other words, considering any $U \subset \mathbb{F}_q^n$ with $\dim(U) = i$, we must be able to rank $\mathbb{F}_q^n \setminus U$ (*outside* a subspace).

Attaching a meaningful order

A first example

The design of (un)ranking functions depends on $\prec$, hence we must choose it carefully. A visualization is often helpful.

A possible order: read the coefficients left to right, top to bottom, then sort the resulting sequence lexicographically.

Below is the proposed ordering for $M_2(\mathbb{F}_2)$ (black is 1, gray is 0).



Attaching a meaningful order

A first example

The design of (un)ranking functions depends on $\prec$, hence we must choose it carefully. A visualization is often helpful.

A possible order: read the coefficients left to right, top to bottom, then sort the resulting sequence lexicographically.

Below is the proposed ordering for $M_2(\mathbb{F}_2)$ (black is 1, gray is 0).



If we focus on the elements of $GL_2(2)$, how do we obtain the r -th invertible matrix? Is this “usual” order useful?

Attaching a meaningful order

Column by column lexicographic

Since we build the matrix column-wise, we can also read entries column-wise, and sort the resulting sequence lexicographically.

Below is $M_2(\mathbb{F}_2)$ in “column-major lexicographic order”.



The matrices now group by the first column.

Attaching a meaningful order

Column by column lexicographic

Since we build the matrix column-wise, we can also read entries column-wise, and sort the resulting sequence lexicographically.

Below is $M_2(\mathbb{F}_2)$ in “column-major lexicographic order”.



The matrices now group by the first column. But within each group, the vectors in the spanned subspace are *not always a contiguous prefix*.

Fix $v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, whose span is $\langle \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rangle = \{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \}$. In lex order, we must choose from $(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix})$, which are at disjoint intervals.

Attaching a meaningful order

Column by column reverse lexicographic

Below is $M_2(\mathbb{F}_2)$ in “column-major *reverse* lexicographic order”.



Attaching a meaningful order

Column by column reverse lexicographic

Below is $M_2(\mathbb{F}_2)$ in “column-major *reverse* lexicographic order”.



In reverse-lex order, if $v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we now choose from $\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right)$; vectors outside the subspace are contiguous.

An additional ingredient: basis completion

The ability to rank vectors relative to a subspace U requires a *deterministic* way to attach an order to $\mathbb{F}_q^n$.

We use a canonical basis completion algorithm $\mathbf{C}_B$ relying on the computation of a row-reduced echelon form (RREF).

Lemma

Let U be a subspace of $V = \mathbb{F}_q^n$ and let $E(U)$ be the basis constructed by $\mathbf{C}_B$. The first $q^{\dim(U)}$ vectors of V under the order $\prec_{E(U)}$ are precisely the vectors in U .

Evaluating coordinates relative to the basis $E(U)$ under the reverse lex order deterministically maps the subspace into the desired prefix, completing the combinatorial tower.

Algorithm descriptions & correctness

Since the evaluation of $\mathbf{R}_V$ relative to $E(U)$ maps U to a contiguous integer prefix, we have that $\mathbf{R}_C(U, v) = \mathbf{R}_V(E(U), v) - q^{\dim(U)}$.

To rank a matrix $M \in \text{GL}_n(q)$, we evaluate its columns sequentially. For each column $i \in \{0, \dots, n-1\}$, let $S_i = (M_{*,0}, \dots, M_{*,i-1})$.

The local rank of the i -th column is an integer in $[N_i]$, where $N_i = q^n - q^i$, and computed as $k_i = \mathbf{R}_C(S_i, M_{*,i})$.

The global rank is the mixed-radix evaluation over these local choices, i.e., $\mathbf{R}_{\text{GL}}(M) = \sum_{i=0}^{n-1} k_i \prod_{j=0}^{i-1} N_j$.

Correctness follows by *induction* on i . Each iteration maps a mixed-radix digit to a valid extension of the subspace, one-to-one.

Time complexity of (un)ranking

The sequential computation of the mixed-radix digits requires evaluating subspace projections at each step.

Proposition

The cost of $\mathbf{R}_{\text{GL}}$ and $\mathbf{U}_{\text{GL}}$ is $\mathcal{O}(n^4 \log q)$ field operations.

It is possible to incrementally update the set of available pivot positions rather than fully recomputing the RREF at each iteration. (The cost is reduced to $\mathcal{O}(n^\omega)$ field operations.)

In practical terms, (un)ranking only needs the linear algebra already present in any implementation that uses rejection sampling.

It is the 2nd fastest sampler (behind Schubert) for $q = 16$ and only $1.06\text{--}1.63\times$ slower than rejection sampling for larger fields.

Empirical measurements of random bit complexity

We consider sampling in several word sizes w (in bits).

We employ the *fast dice roller* algorithm [Lumbroso, 2013] to reduce wasted random bits when sampling is not aligned to a machine word.

n	q	w	bruhat	nonsing	rej	schubert	subgroup	unrank
5	16	1	3.00	2.00	2.13	3.00	3.00	2.00
		4	4.92	3.00	4.18	4.00	3.02	2.00
		16	8.05	5.00	7.52	6.98	6.18	2.00
		64	29.00	15.00	26.86	23.75	19.00	2.00
11	65521	1	46.04	32.00	33.00	44.00	34.03	31.00
		4	52.00	35.00	38.00	49.00	38.00	31.00
		16	76.00	46.00	61.00	70.01	51.00	31.00
		64	139.00	76.00	121.00	128.00	86.00	31.00
30	2039	1	245.08	162.85	169.81	239.39	168.07	155.00
		4	285.29	183.12	208.94	278.31	190.02	155.00
		16	441.60	260.06	360.74	429.58	271.31	155.00
		64	989.21	530.25	900.00	959.23	559.22	155.00

Next steps & acknowledgements

Attacking $GL_n(q)$ solves (un)ranking for $Aff_n(q)$, $SL_n(q)$, $PGL_n(q)$ etc.; we are expanding to other (not necessarily linear) groups.

We are also formalizing the algorithm descriptions and inductive proofs in the Lean proof assistant.

[Gregor et al., 2024] gives an enumeration algorithm considering a Gray order. [Narayanan, 2025] gives a sampling algorithm for random tensors. Can these be extended to (un)ranking?

Partially supported by FAPESC (61/2024, 62/2024), ON-RCPN of Brazil, and NSERC (RGPIN-2024-05341).

Thank you! Any questions?

References I



Diaconis, P. and Shahshahani, M. (1987).

The Subgroup Algorithm for Generating Uniform Random Variables.

Probability in the Engineering and Informational Sciences,
1(1):15–32.



Gregor, P., Hoang, H. P., Merino, A., and Mička, O. (2024).

Generating All Invertible Matrices by Row Operations.

In Mestre, J. and Wirth, A., editors, *35th International Symposium on Algorithms and Computation (ISAAC 2024)*, volume 322 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:14.

References II



Lumbroso, J. (2013).

Optimal Discrete Uniform Generation from Coin Flips, and Applications.

arXiv:1304.1916v1.



Narayanan, A. K. (2025).

Strong Keys for Tensor Isomorphism Cryptography.

In Gawrychowski, P., Mazowiecki, F., and Skrzypczak, M., editors, *50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025)*, volume 345 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 78:1–78:18, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

References III



Pak, I. (1997).

When and how n choose k .

In Pardalos, P. M., Rajasekaran, S., and Rolim, J., editors, *Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–238.



Randall, D. (1993).

Efficient generation of random nonsingular matrices.

Random Structures & Algorithms, 4(1):111–118.